Session Code: TLS300

tools & languages

Visual Basic “Whidbey”: Advanced Language and IDE Features

Steven Lees,
stevelee@microsoft.com
Amanda Silver,
amandas@microsoft.com
Program Managers
Microsoft Corporation

PDC 03
Make the connection

Microsoft

Agenda

- Advanced IDE Features
- Lots of new language features
- Full platform support

Advanced IDE Features

- Application Designer
  - Settings
- Extensible code snippets
- Rename Symbol
- New Async calling convention

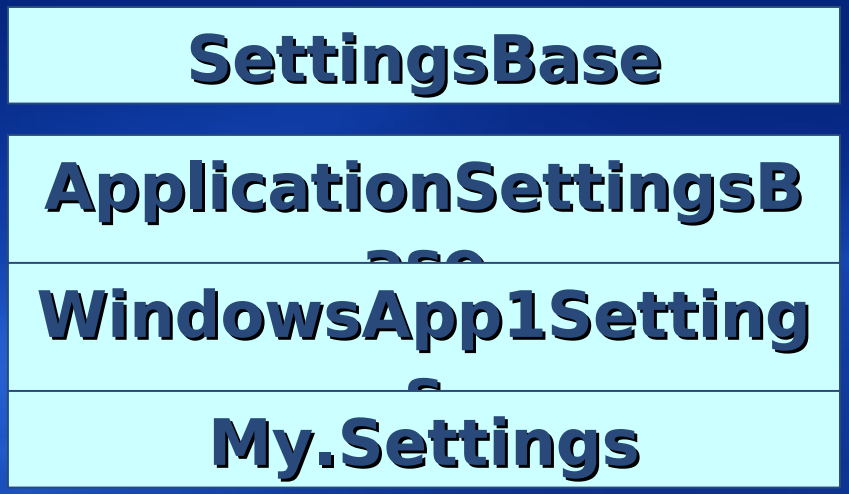
Configuration and Settings

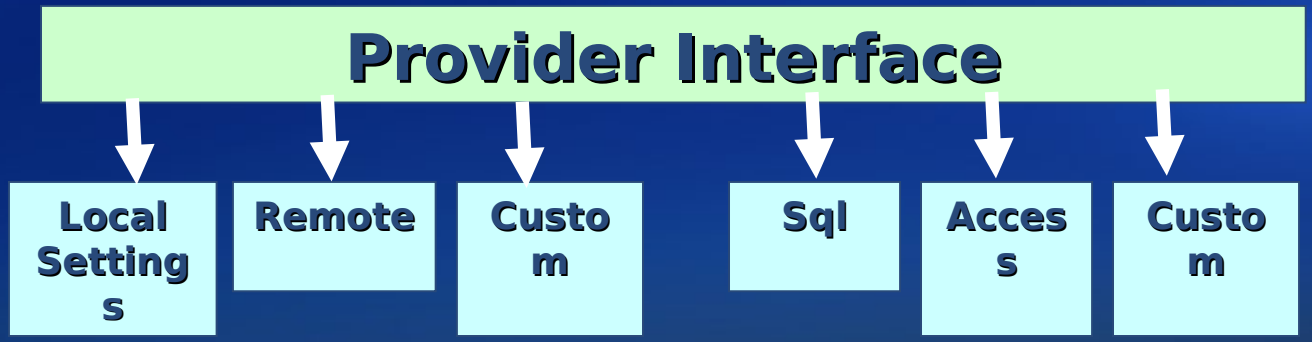
- Basic support in VS 2002 and 2003
  - App.config contains settings in XML
  - Framework class for reading settings
  - Can extend with additional capabilities
    - Straightforward, but not really easy

Configuration and Settings

- Huge improvements in Whidbey
  - Framework classes support read/write
  - Strongly-typed access to settings
  - IntelliSense for settings
  - Choose app-scoped or user-scoped settings
  - Works in partial trust
  - Validation model
  - Extensible provider model
  - Shared infrastructure across client and web

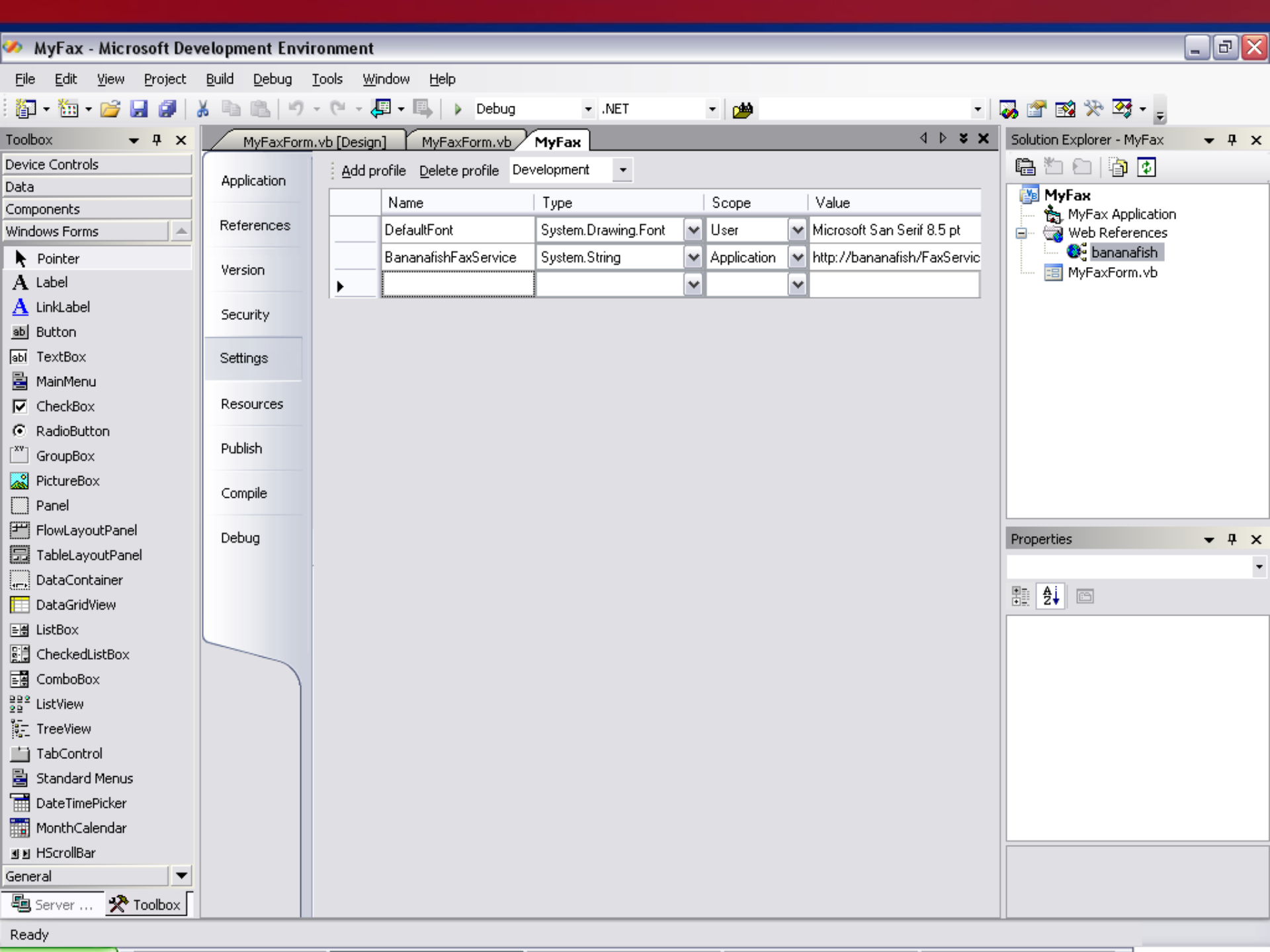
Settings Architecture





Settings Architecture

- Customizable
  - At the provider level
  - By writing custom settings classes


MyFax - Microsoft Development Environment
File Edit View Project Build Debug Tools Window Help
Debug .NET
Toolbox
Device Controls
Data
Components
Windows Forms
Pointer
Label
LinkLabel
Button
TextBox
MainMenu
CheckBox
RadioButton
GroupBox
PictureBox
Panel
FlowLayoutPanel
TableLayoutPanel
DataContainer
DataGridView
ListBox
CheckedListBox
ComboBox
ListView
TreeView
TabControl
Standard Menus
DateTimePicker
MonthCalendar
HScrollBar
General
Server ...
Toolbox
MyFaxForm.vb [Design]
MyFaxForm.vb
MyFax
Application
References
Version
Security
Settings
Resources
Publish
Compile
Debug
Add profile
Delete profile
Development
Name
Type
Scope
Value
DefaultFont
System.Drawing.Font
User
Microsoft San Serif 8.5 pt
BananafishFaxService
System.String
Application
http://bananafish/FaxServic
Solution Explorer - MyFax
MyFax
MyFax Application
Web References
bananafish
MyFaxForm.vb
Properties
Ready

Settings

- Designer allows
  - Creating / modifying / deleting entries
  - Type of entry (for strong typing)
  - Choosing user-scoped or app-scoped
  - Default value
  - Different “profiles”
- Special support for
  - Connection Strings
  - Web service proxies
  - VS will automatically create settings

Settings Architecture

- App settings

Myapp.exe.config

<applicationSettings>
...
</applicationSettings>

- User Settings

fred.config

ethel.config

gladys.config

<userSettings>
...
</userSettings>

Code Samples

- Load

```
My.Settings.UseHighQuality = True
‘ Settings automatically loaded on first access
```

- Save

```
My.Settings.UseHighQuality = True
My.Settings.Save()
```

- Handling events for validation

```
Private Sub Settings_SettingChanging(ByVal sender As Object, ByVal e As
SettingsArg) Handles MyBase.SettingChanging

    If e.SettingName = “SignatureFile” Then
        If Not My.Computer.FileSystem.FileExists(e.Setting.Value) Then
            ‘ Cancel event
        End If
    End If
End Sub
```

Advanced IDE Features

- Application Designer
  - Settings
- Extensible code snippets
- Rename Symbol
- New Async calling convention

Application Designer

- One stop shopping for app settings
  - Resources
  - App and user settings
  - References
  - Version Info
  - Deployment
- Easier to find
- Non-modal window for better usability

Application Designer

- New Application Events
  - Startup
  - Shutdown
  - Unhandled exception
  - Network connected
  - Network disconnected

Advanced IDE Features

- Application Designer
  - Settings
- Extensible code snippets
- Rename Symbol
- New Async calling convention

Extensible Code Snippets

- Code snippets have standard schema
- You can add items to the right-click menu
- Snippet editor
  - For creating, editing snippets
- Multiple snippet stores
  - Local or network share
- Easy way to package code for reuse
- Sharing is encouraged!

Rename Symbol
Quick way to change identifiers

- Rename all cases of “TextBox1”
  - Type declaration
  - Variable declarations
  - Events
- More precise than text replacement
- Doesn’t affect comments
- Working on other “refactoring” features
  - Create method

Advanced IDE Features

- Application Designer
  - Settings
- Extensible code snippets
- Rename Symbol
- New Async calling convention

Asynchronous calling

- Easier way to call slow background tasks
- Model asynchronous calls as events
  - PictureBox1.LoadAsync(url)
  - PictureBox1_LoadCompleted(..., ...)
- Completion routine is on main thread
- Web service proxies
- Supports progress, cancellation
- Background worker component

Background Worker

```
Structure BackgroundArgs
    Dim arg1 As Integer
    Dim arg2 As String
End Structure

Private Sub Button1_Click(…) Handles Button1.Click
    Dim args As BackgroundArgs
    BackgroundWorker1.RunWorkerAsync(args)
End Sub

Private Sub BackgroundWorker1_DoWork(…) Handles
BackgroundWorker1.DoWork
    ‘ This is run on a background thread
    Dim args As BackgroundArgs
    args = CType(e.Argument, BackgroundArgs)
    ‘ Now do something on the background thread
    e.Result = <some object>
End Sub

Private Sub BackgroundWorker1_RunWorkerCompleted(…) Handles
Background…
    result = e.Result
End Sub
```

Advanced IDE Features

demo

PDC03

Make the connection

New Language Features

- Explicit array bounds
- Using statement
- Continue statement
- Global keyword
- Accessor accessibility
- Partial types
- Unsigned types
- Operator Overloading
- Generics
- Warnings

Explicit Array Bounds

- Arrays are still zero based
- You can now specify the lower and upper bounds in the declaration

```
‘In VS 2002 and 2003, you did it this way
Dim x(10) As Integer

‘Now you can also declare it like this
Dim y(0 To 10) As Integer
```

Using Statement
Acquire, Execute, Release

- Fast way correctly release resources
  - Easier to read than Try, Catch, Finally
- Couple with Dispose Pattern stub gen

```
‘Using block disposes of resource
Using fStr As New FileStream(path, FileMode.Append)

  For i As Integer = 0 To fStr.Length
      fStr.ReadByte()
  Next

  ‘End of block will close stream
End Using
```

Continue Statement
Skips to next iteration of loop

- Loop logic is concise, easier to read

```
For j As Integer = 0 to 5000
  While matrix(j) IsNot thisValue
    If matrix(j) Is thatValue

      ‘ Continue to next j
      Continue For

    End If

    Graph(j)
  End While
Next j
```

Global Keyword
Access to root (empty) namespace

- Complete name disambiguation
- Better choice for code generation

```
Namespace HeadTrax
  Class Form1
     Inherits Windows.Forms.Forms

  Sub LastName(nm As String)
    Global.Microsoft.VisualBasic.Left(nm)
  End Sub

  End Class
End Namespace
```

Accessor Accessibility

Granular accessibility on Get and Set

- Forces calls to always use get, set
- Easy way to enforce validation

```
Property Salary() As Integer
  Get
     Return mSalary
  End Get

  Private Set( value As Integer)
     If value < 0 Then
       Throw New Exception(“Mistake”)
     End If
  End Set
End Property
```

Partial Types

Single Structure, Class in Multiple Files

- Used to separate designer gen into another file
- Easier implementation

```
Public Class Form1
  Inherits Windows.Forms.Form
     ‘ Your Code
End Class

Partial Class Form1
     ‘ Designer code
     Sub InitializeComponent()
       ‘ Your controls
     End Sub
End Class
```

Unsigned Types
Full support in the language

- Full platform parity
- Easier Win32 Api calls and translation
- Memory and performance win

```
Dim sb As SByte = -4 ‘Error:negative
Dim us As UShort
Dim ui As UInteger
Dim ul As ULong

‘Full support in VisualBasic modules
If IsNumeric(uInt) Then
     ‘ Will now return true
End If
```

New Language Features

demo

PDC03

Make the connection

Operator Overloading
Create your own base types

```
Class Addr
  Private mString As String

  Property Value() As String
    Get
      Return mString
    End Get
    Set (value As String)
      If Valid(value) Then
        mString = value
      End If
    End Set

  Shared Operator &(ad1 As Addr, ad2 As Addr) _
    As Addr

    Return New Addr(ad1.Value & ad2.Value)
  End Operator
```

Operator Overloading

demo

PDC03

Make the connection

Generics

```
Public Class List
   Private elements() As Object
   Private mCount As Integer

   Public Sub Add(element As Object)
      If (mCount = elements.Length) Then _
        Resize(mCount * 2)
      mCount += 1
      elements(mCount) = element
   End Sub

   Default Public Property i(index As Integer) As Object
      Get
        Return elements(index)
      End Get
      Set
        E
   End
   Publ
      G
   End
End Cla
```

```
Dim intList As New List(Of Integer)

intList.Add(1)              ‘ No boxing
intList.Add(2)              ‘ No boxing
intList.Add("Three")        ‘ Compile-time error

int i = intList(0)          ‘ No cast required
```

Generics (Specifics)

- Compile Time Checking
  - Eliminates runtime errors
- Performance
  - No casting or boxing overhead
- Code reuse
  - Easy to create strongly typed collections
- Generic Collection classes in Framework
  - Dictionary, HashTable, List, Stack, etc.

Generics

demo

PDC03

Make the connection

Visual Basic Warnings
Early warning of bad runtime behavior

- Overlapping catch blocks or cases
- Recursive property access
- Unused Imports statement
- Unused local variable
- Function, operator without return
- Reference on possible null reference
- Option Strict broken down
  - “Late binding”
  - VB Style Conversions
  - Etc.

Warnings

demo

PDC03

Make the connection

Additional Resources

- Related sessions
  - TLS343 – Visual Studio “Whidbey”: Advanced Debugging Techniques (Tuesday 5:15 Room 408 AB)
  - TLS344 – Visual Studio “Whidbey”: Deploying Applications Using ClickOnce (Wednesday 10:00 Room 511 ABC)
  - ARC413 – CLR Under the Covers: “Whidbey” CLR Internals (Wednesday 5:00 Room 152/153)

Additional Resources

- Panels
  - Ask The Experts (Tonight)
  - Languages Panel (Thursday)
- Send us feedback!
  - vswish@microsoft.com
  - Newsgroup: microsoft.public.dotnet.vb
  - http://msdn.microsoft.com/vstudio/whidbeypdc
- Other resources
  - Paul Vick's blog: www.panopticon.net

Summary

- Ease of use is built on solid infrastructure
- Many points of extensibility
- Very powerful new language features
  - And some cool fit and finish ones too
- Full access to the platform

- Please fill out the eval!
- (Thank you!)

Make the connection

Microsoft Professional Developers Conference 2003

October 26 - 30, 2003, Los Angeles, CA

Microsoft